

# CHAPTER 4:

## LOGICAL DATABASE DESIGN AND THE RELATIONAL MODEL

*Modern Database Management*

*11<sup>th</sup> Edition*

*Jeffrey A. Hoffer, V. Ramesh,  
Heikki Topi*

---

# OBJECTIVES

---

- ✗ Define terms
- ✗ List five properties of relations
- ✗ State two properties of candidate keys
- ✗ Define first, second, and third normal form
- ✗ Describe problems from merging relations
- ✗ Transform E-R and EER diagrams to relations
- ✗ Create tables with entity and relational integrity constraints
- ✗ Use normalization to convert anomalous tables to well-structured relations

# COMPONENTS OF RELATIONAL MODEL

---

- ✗ Data structure
  - + Tables (relations), rows, columns
- ✗ Data manipulation
  - + Powerful SQL operations for retrieving and modifying data
- ✗ Data integrity
  - + Mechanisms for implementing business rules that maintain integrity of manipulated data



# RELATION

---

- ✗ A relation is a named, two-dimensional table of data.
- ✗ A table consists of rows (records) and columns (attribute or field).
- ✗ Requirements for a table to qualify as a relation:
  - + It must have a unique name.
  - + Every attribute value must be atomic (not multivalued, not composite).
  - + Every row must be unique (can't have two rows with exactly the same values for all their fields).
  - + Attributes (columns) in tables must have unique names.
  - + The order of the columns must be irrelevant.
  - + The order of the rows must be irrelevant.

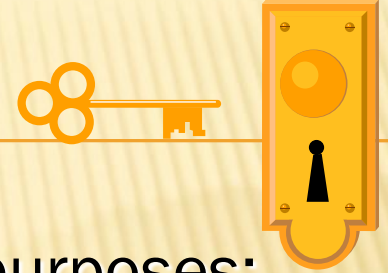
**NOTE: All *relations* are in *1<sup>st</sup> Normal form*.**



# CORRESPONDENCE WITH E-R MODEL

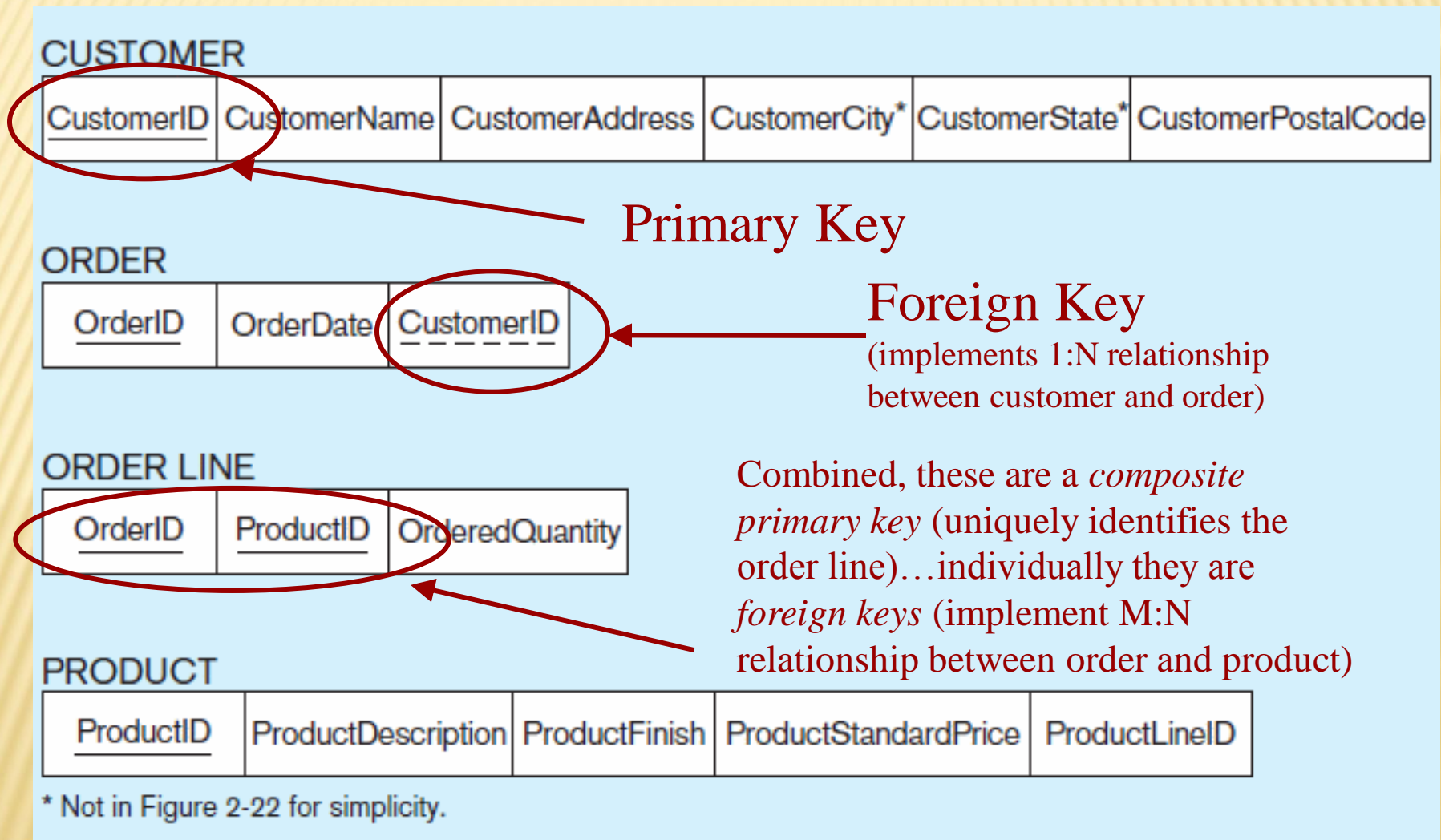
- ✗ Relations (tables) correspond with entity types and with many-to-many relationship types.
- ✗ Rows correspond with entity instances and with many-to-many relationship instances.
- ✗ Columns correspond with attributes.
- ✗ NOTE: The word ***relation*** (in relational database) is NOT the same as the word ***relationship*** (in E-R model).

# KEY FIELDS



- ✗ Keys are special fields that serve two main purposes:
  - + **Primary keys** are unique identifiers of the relation. Examples include employee numbers, social security numbers, etc. *This guarantees that all rows are unique.*
  - + **Foreign keys** are identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship).
- ✗ Keys can be **simple** (a single field) or **composite** (more than one field).
- ✗ Keys usually are used as indexes to speed up the response to user queries (more on this in Chapter 5).

Figure 4-3 Schema for four relations (Pine Valley Furniture Company)





# INTEGRITY CONSTRAINTS

---

- ✖ Domain Constraints
  - + Allowable values for an attribute (See Table 4-1)
- ✖ Entity Integrity
  - + No primary key attribute may be null. All primary key fields **MUST** have data.
- ✖ Action Assertions
  - + Business rules (Recall from Chapter 3)

**TABLE 4-1 Domain Definitions for INVOICE Attributes**

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

**Domain definitions enforce domain integrity constraints.**

# INTEGRITY CONSTRAINTS

- ✗ **Referential Integrity**—rule states that any foreign key value (on the relation of the many side) **MUST** match a primary key value in the relation of the one side. (Or the foreign key can be null)
  - + For example: Delete Rules
    - ✗ **Restrict**—don't allow delete of “parent” side if related rows exist in “dependent” side
    - ✗ **Cascade**—automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted
    - ✗ **Set-to-Null**—set the foreign key in the dependent side to null if deleting from the parent side → not allowed for weak entities



Figure 4-5  
Referential integrity constraints (Pine Valley Furniture)

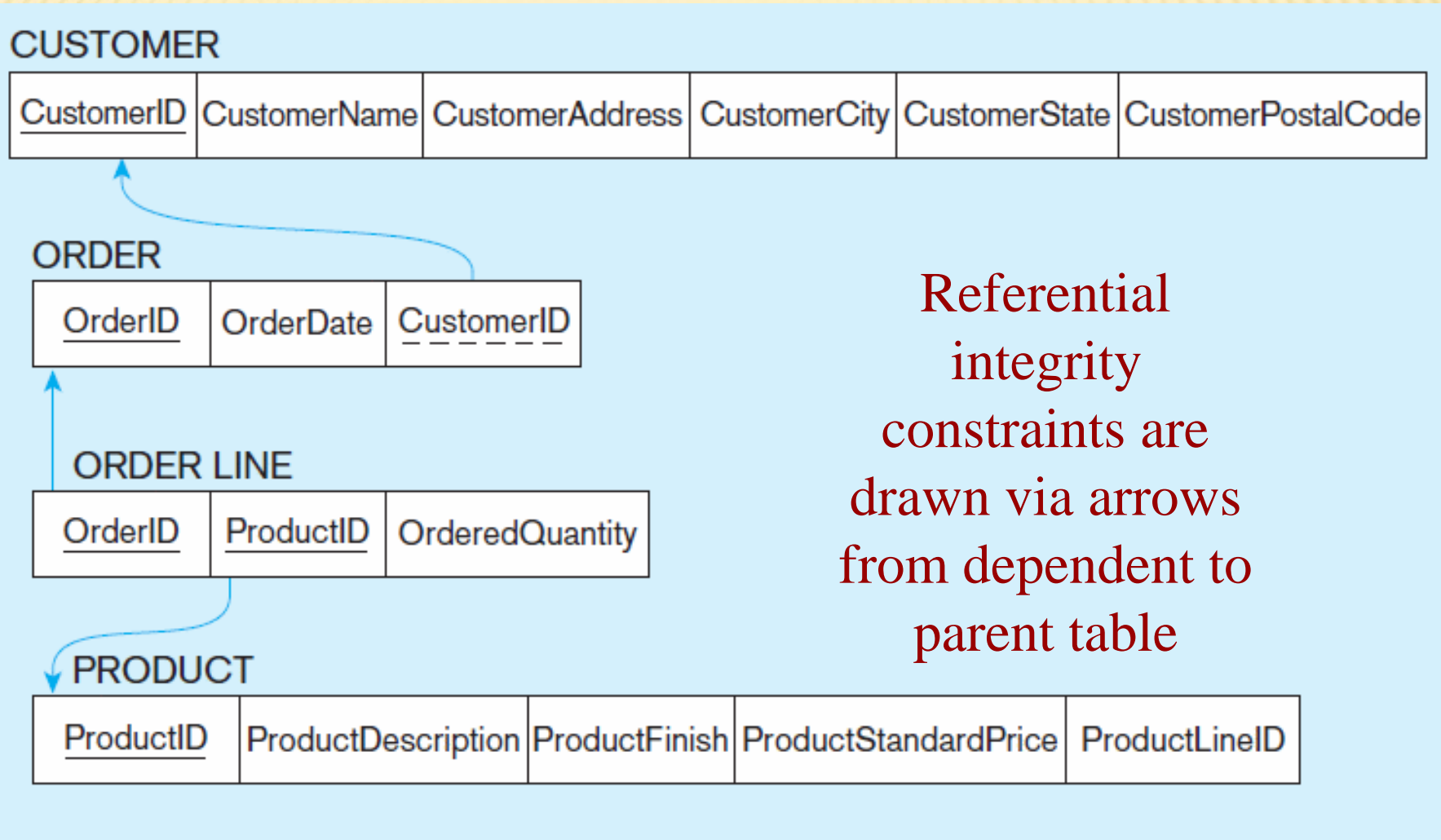


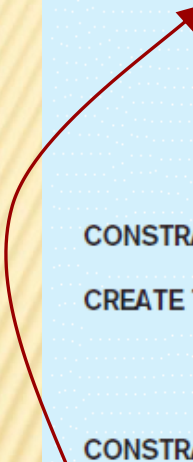
Figure 4-6 SQL table definitions

```
CREATE TABLE Customer_T
  (CustomerID          NUMBER(11,0)    NOT NULL,
   CustomerName        VARCHAR2(25)    NOT NULL,
   CustomerAddress     VARCHAR2(30),
   CustomerCity        VARCHAR2(20),
   CustomerState       CHAR(2),
   CustomerPostalCode  VARCHAR2(9),
  CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
  (OrderID             NUMBER(11,0)    NOT NULL,
   OrderDate           DATE DEFAULT SYSDATE,
   CustomerID          NUMBER(11,0),
  CONSTRAINT Order_PK PRIMARY KEY (OrderID),
  CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));

CREATE TABLE Product_T
  (ProductID          NUMBER(11,0)    NOT NULL,
   ProductDescription  VARCHAR2(50),
   ProductFinish       VARCHAR2(20),
   ProductStandardPrice DECIMAL(6,2),
   ProductLineID       NUMBER(11,0),
  CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
  (OrderID             NUMBER(11,0)    NOT NULL,
   ProductID           NUMBER(11,0)    NOT NULL,
   OrderedQuantity     NUMBER(11,0),
  CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
  CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
  CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T (ProductID));
```



Referential  
integrity  
constraints are  
implemented with  
foreign key to  
primary key  
references.

# TRANSFORMING EER DIAGRAMS INTO RELATIONS

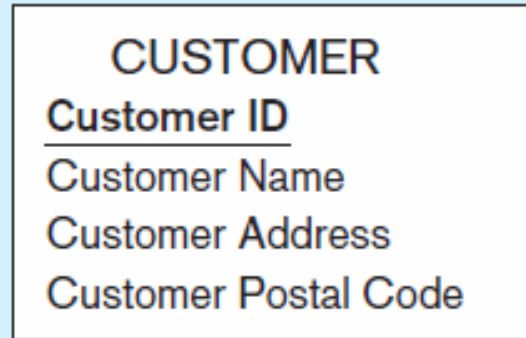
## Mapping Regular Entities to Relations

- + Simple attributes: E-R attributes map directly onto the relation
- + Composite attributes: Use only their simple, component attributes
- + Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity

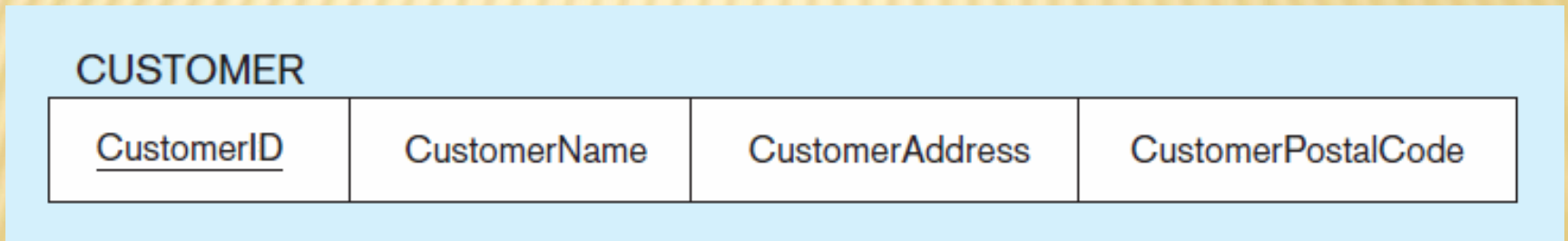


Figure 4-8 Mapping a regular entity

**(a) CUSTOMER  
entity type with  
simple  
attributes**

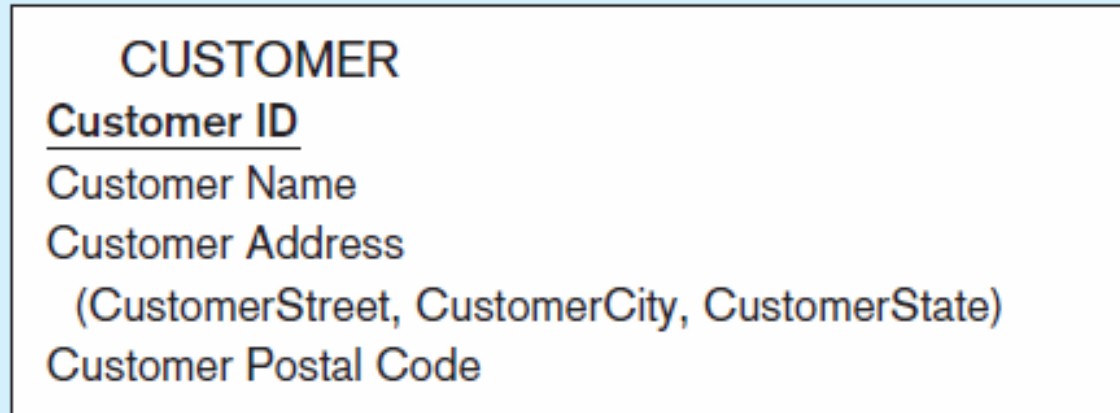


**(b) CUSTOMER relation**



## Figure 4-9 Mapping a composite attribute

(a) CUSTOMER  
entity type with  
composite  
attribute



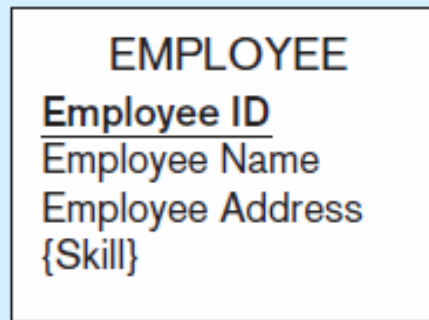
(b) CUSTOMER relation with address detail

CUSTOMER

<u>CustomerID</u>	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode
-------------------	--------------	----------------	--------------	---------------	--------------------

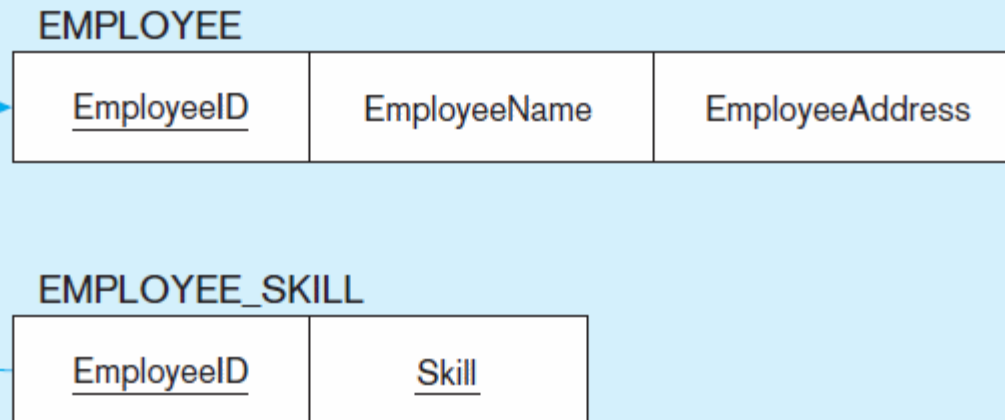
Figure 4-10 Mapping an entity with a multivalued attribute

(a)



**Multivalued attribute becomes a separate relation with foreign key**

(b)



**One-to-many relationship between original entity and new relation**



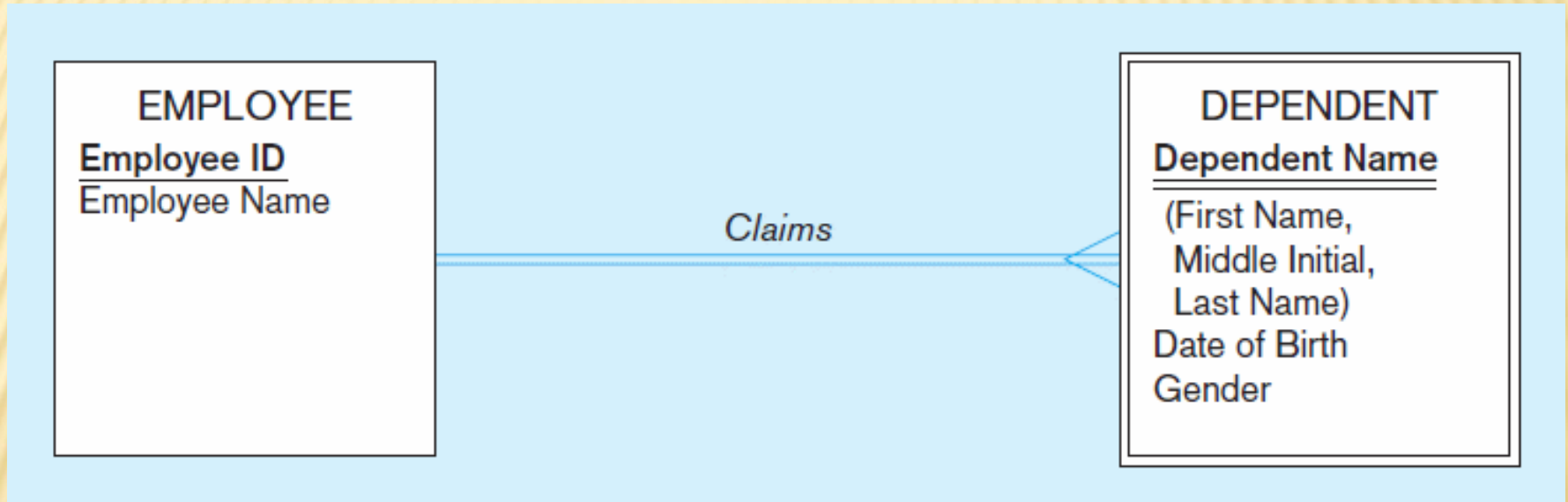
# TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Weak Entities

- + Becomes a separate relation with a foreign key taken from the superior entity
- + Primary key composed of:
  - × Partial identifier of weak entity
  - × Primary key of identifying relation (strong entity)

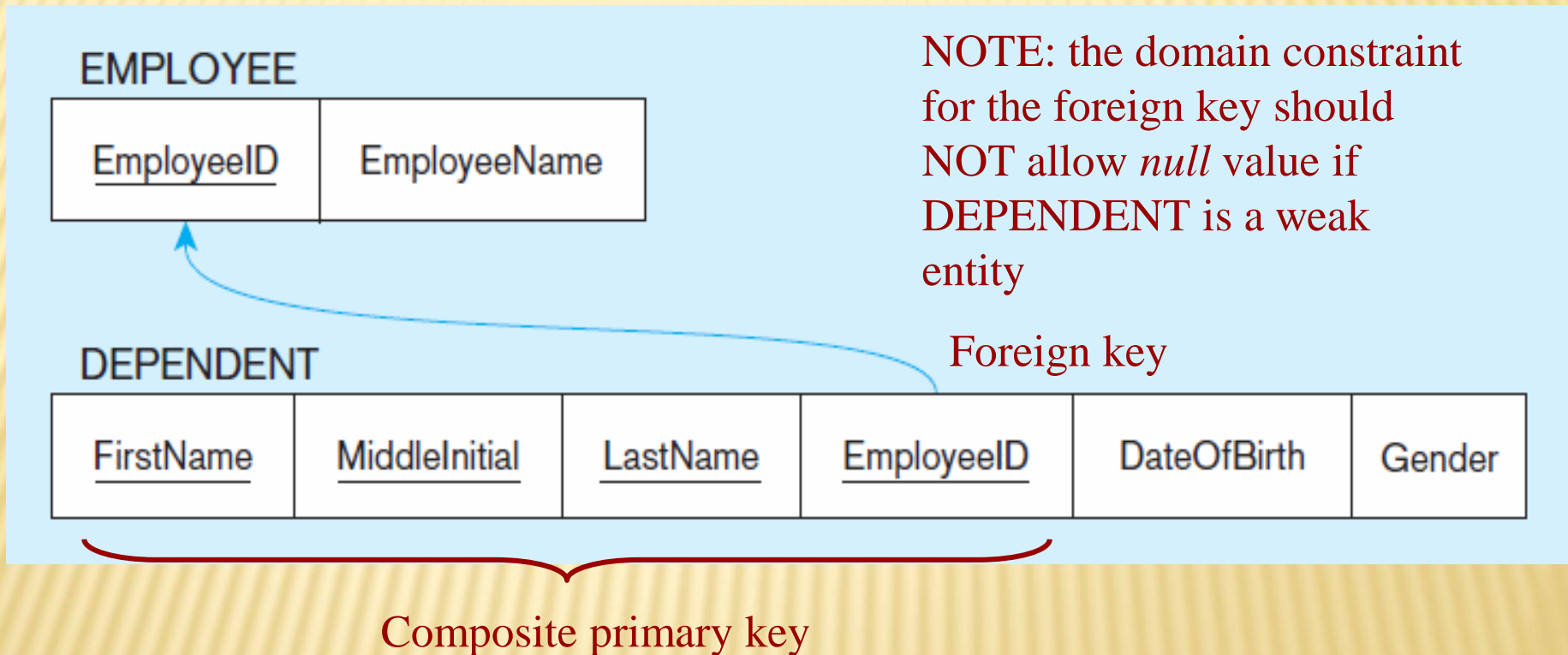
Figure 4-11 Example of mapping a weak entity

a) Weak entity DEPENDENT



## Figure 4-11 Example of mapping a weak entity (cont.)

### b) Relations resulting from weak entity





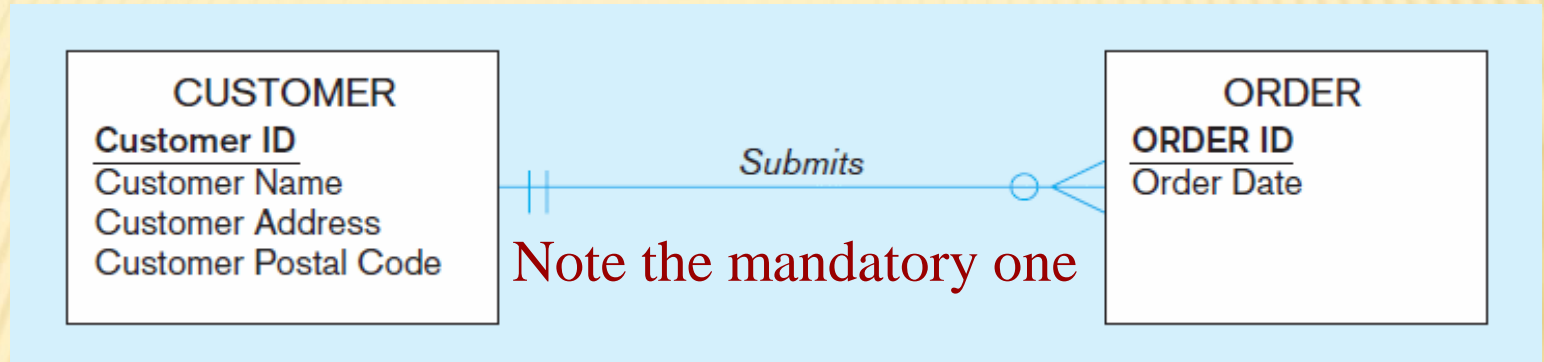
# TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Binary Relationships

- + **One-to-Many**–Primary key on the one side becomes a foreign key on the many side
- + **Many-to-Many**–Create a *new relation* with the primary keys of the two entities as its primary key
- + **One-to-One**–Primary key on mandatory side becomes a foreign key on optional side

## Figure 4-12 Example of mapping a 1:M relationship

### a) Relationship between customers and orders



### b) Mapping the relationship

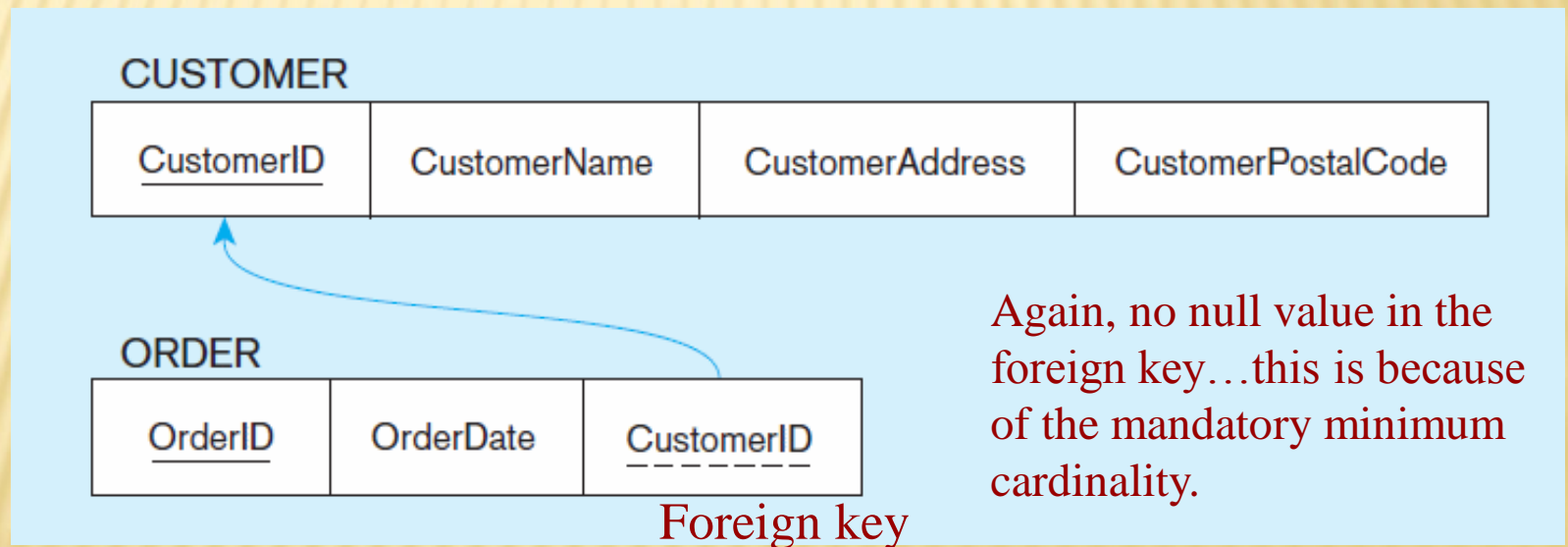
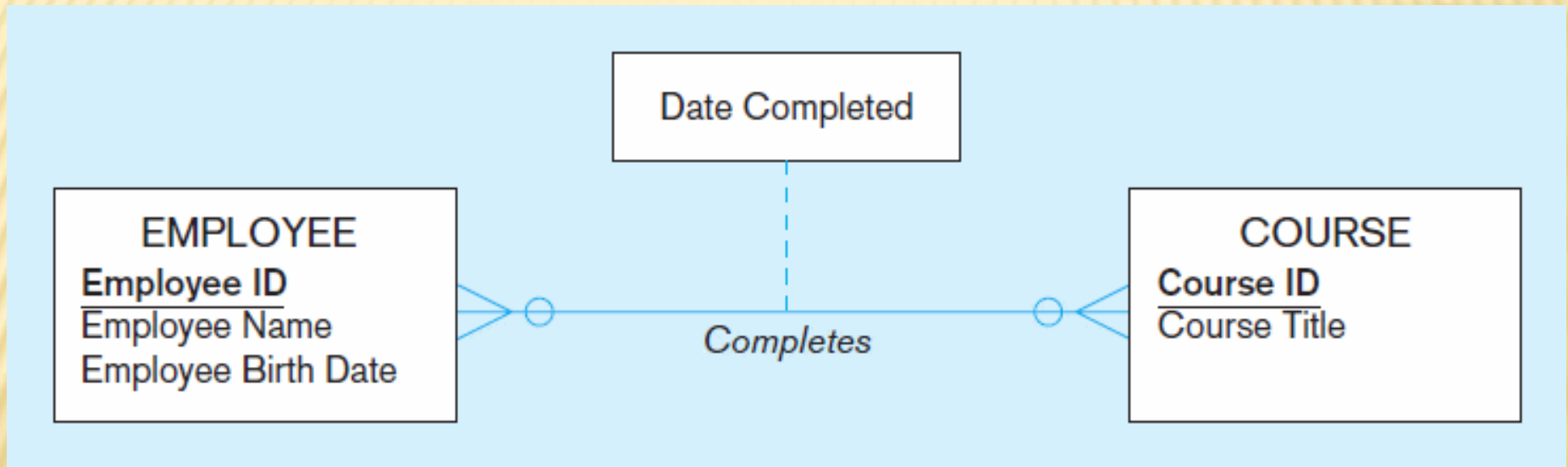


Figure 4-13 Example of mapping an M:N relationship

a) Completes relationship (M:N)



The *Completes* relationship will need to become a separate relation.



Figure 4-13 Example of mapping an M:N relationship (cont.)

b) Three resulting relations

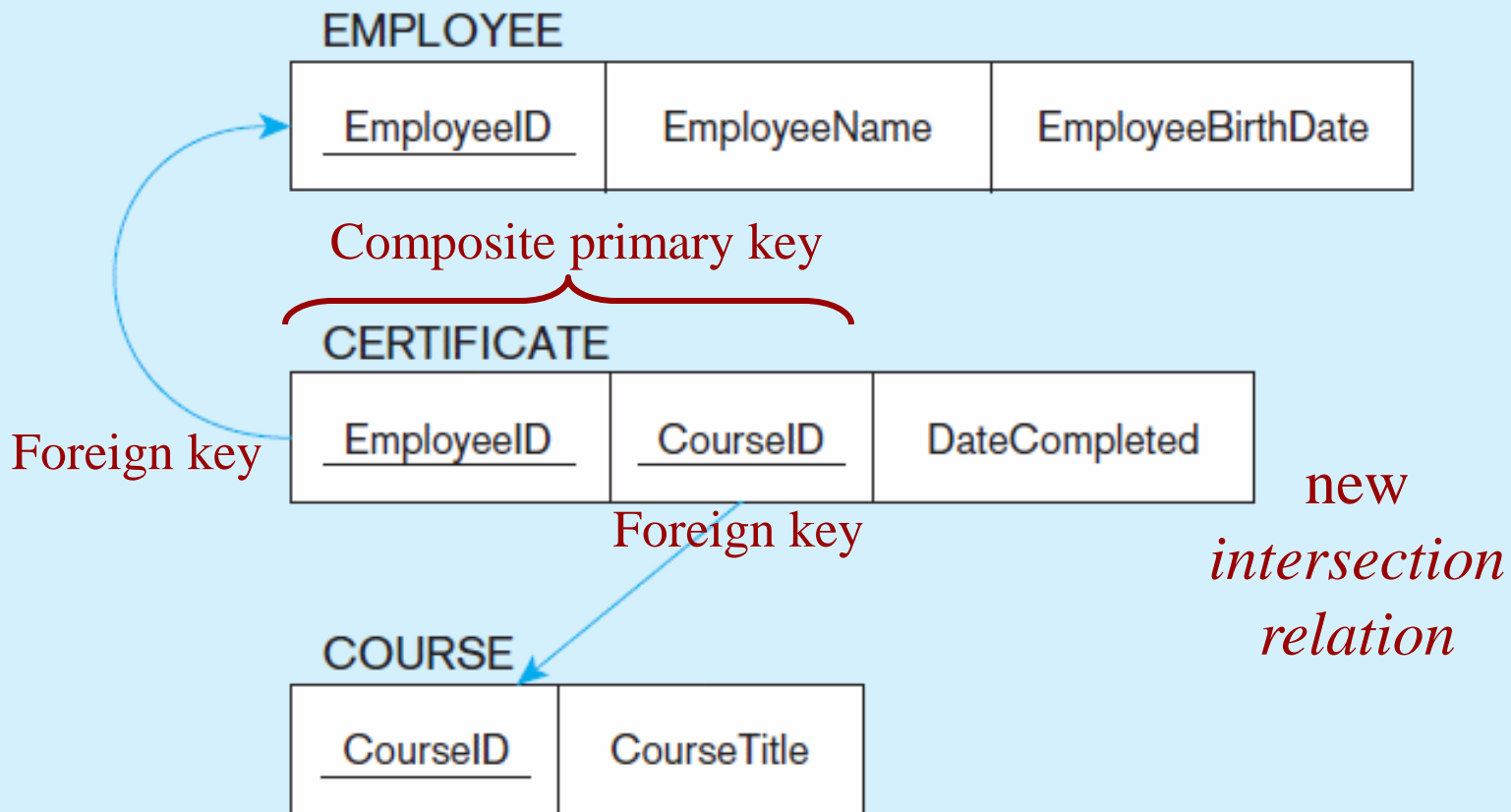
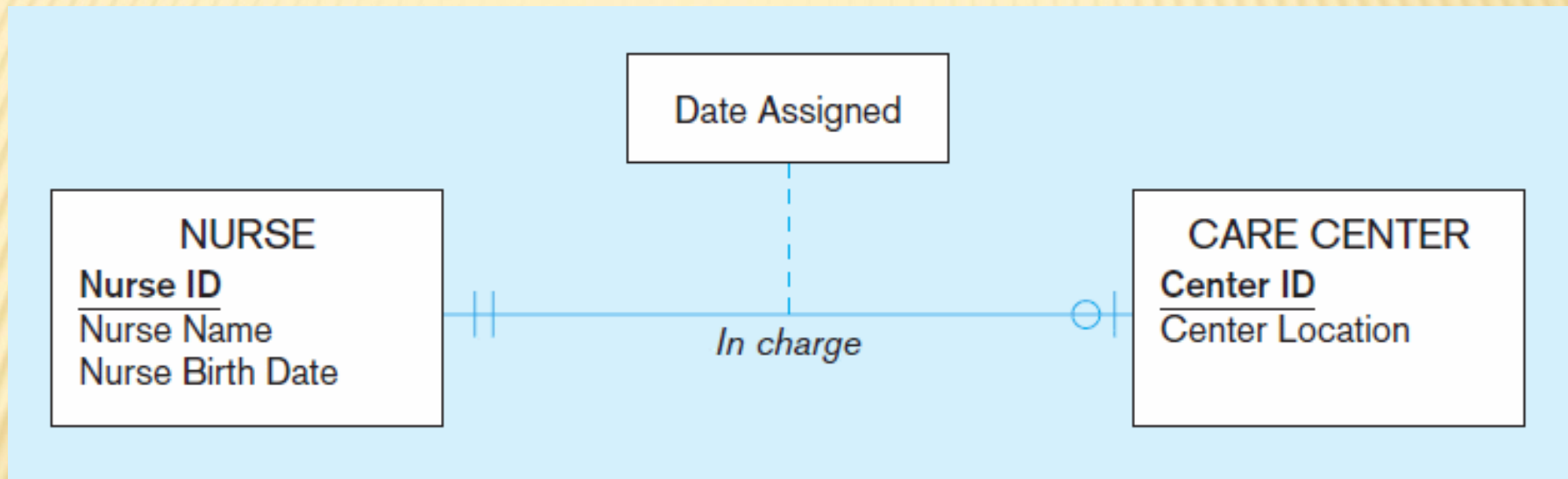


Figure 4-14 Example of mapping a binary 1:1 relationship

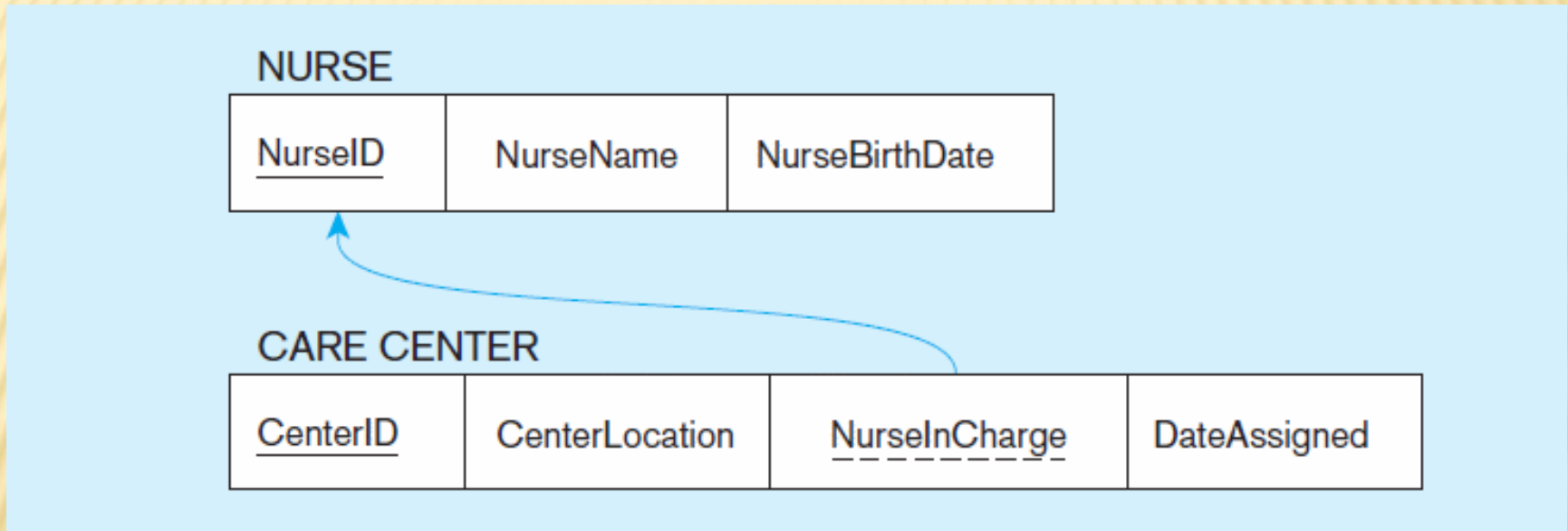
a) In charge relationship (1:1)



Often in 1:1 relationships, one direction is optional

Figure 4-14 Example of mapping a binary 1:1 relationship (cont.)

b) Resulting relations



Foreign key goes in the relation on the optional side, matching the primary key on the mandatory side



# TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Associative Entities

### +Identifier Not Assigned

- ✗ Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)

### +Identifier Assigned

- ✗ It is natural and familiar to end-users
- ✗ Default identifier may not be unique

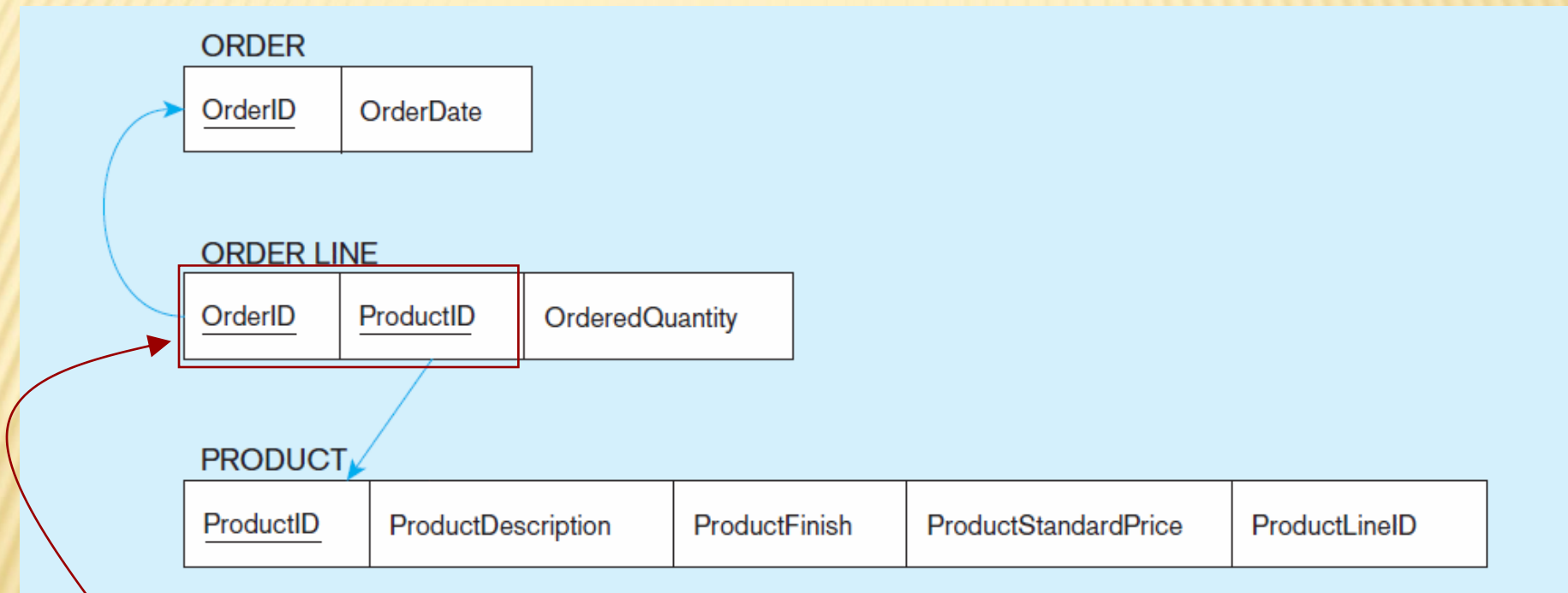
## Figure 4-15 Example of mapping an associative entity

### a) An associative entity



## Figure 4-15 Example of mapping an associative entity (cont.)

### b) Three resulting relations



Composite primary key formed from the two foreign keys



Figure 4-16 Example of mapping an associative entity with an identifier

a) SHIPMENT associative entity

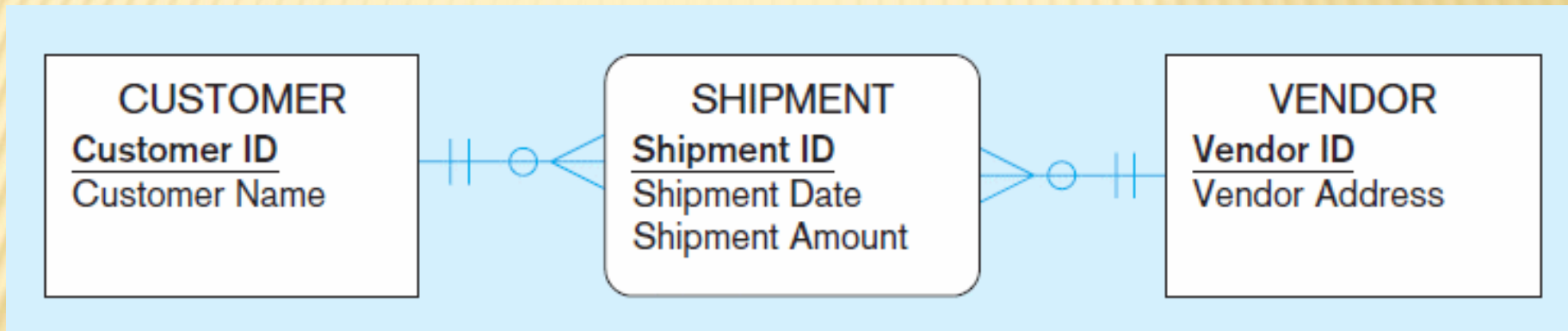
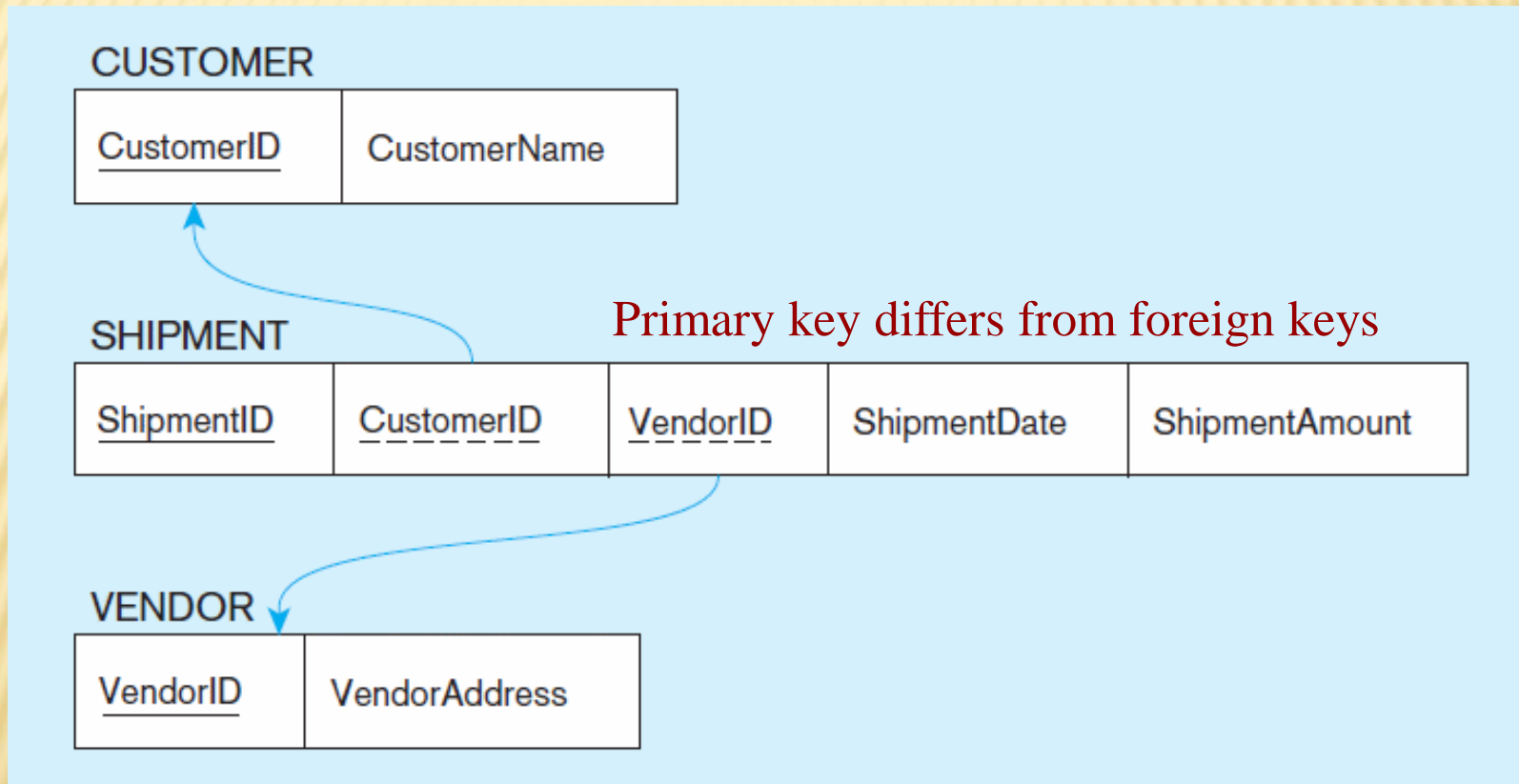


Figure 4-16 Example of mapping an associative entity with an identifier (cont.)

b) Three resulting relations



# TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

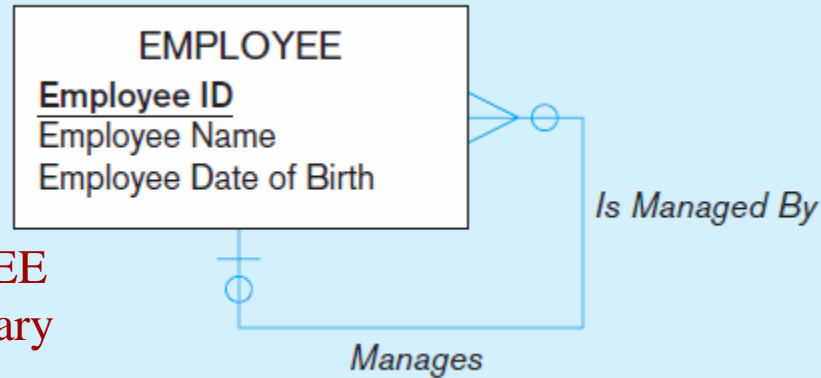
## Mapping Unary Relationships

- + One-to-Many–Recursive foreign key in the same relation
- + Many-to-Many–Two relations:
  - × One for the entity type
  - × One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity



# Figure 4-17 Mapping a unary 1:N relationship

(a) EMPLOYEE  
entity with unary  
relationship



(b)  
EMPLOYEE  
relation with  
recursive  
foreign key

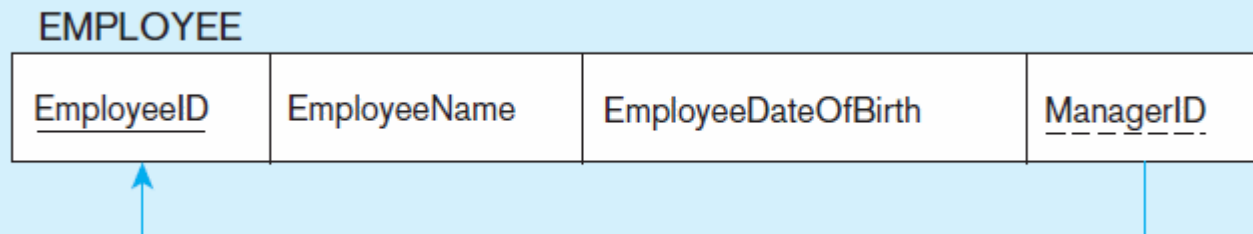
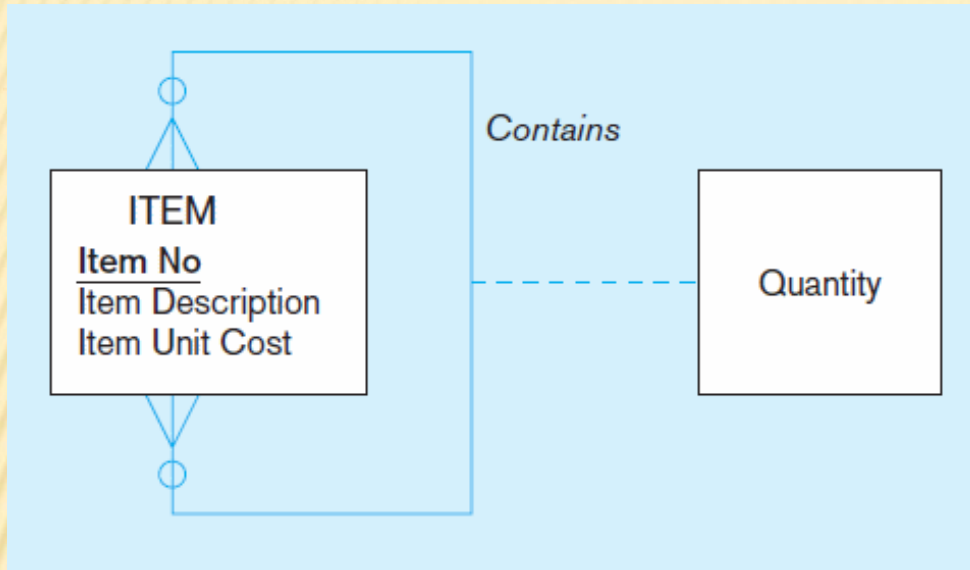
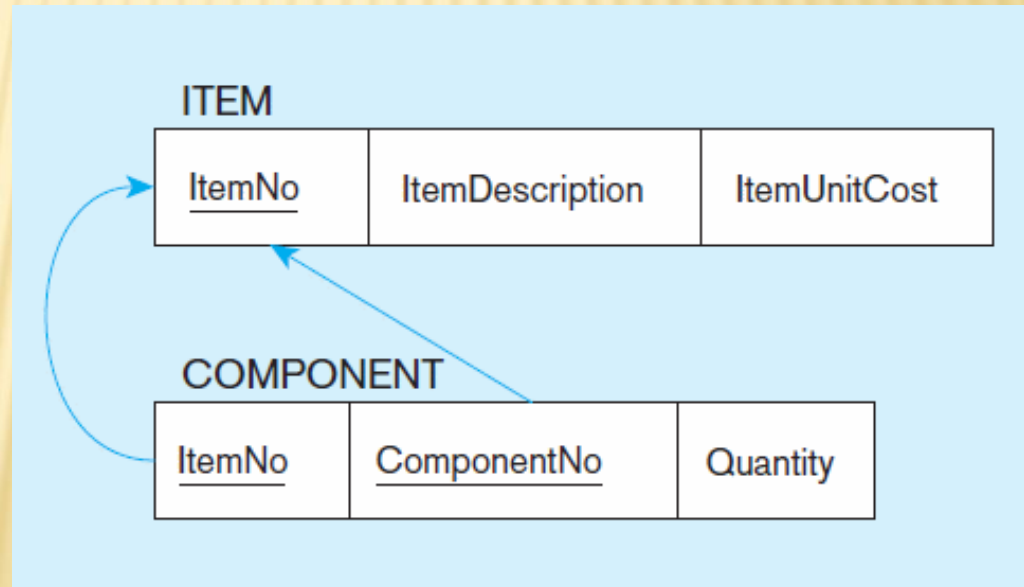


Figure 4-18 Mapping a unary M:N relationship



(a) Bill-of-materials relationships (M:N)

(b) ITEM and COMPONENT relations



# TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

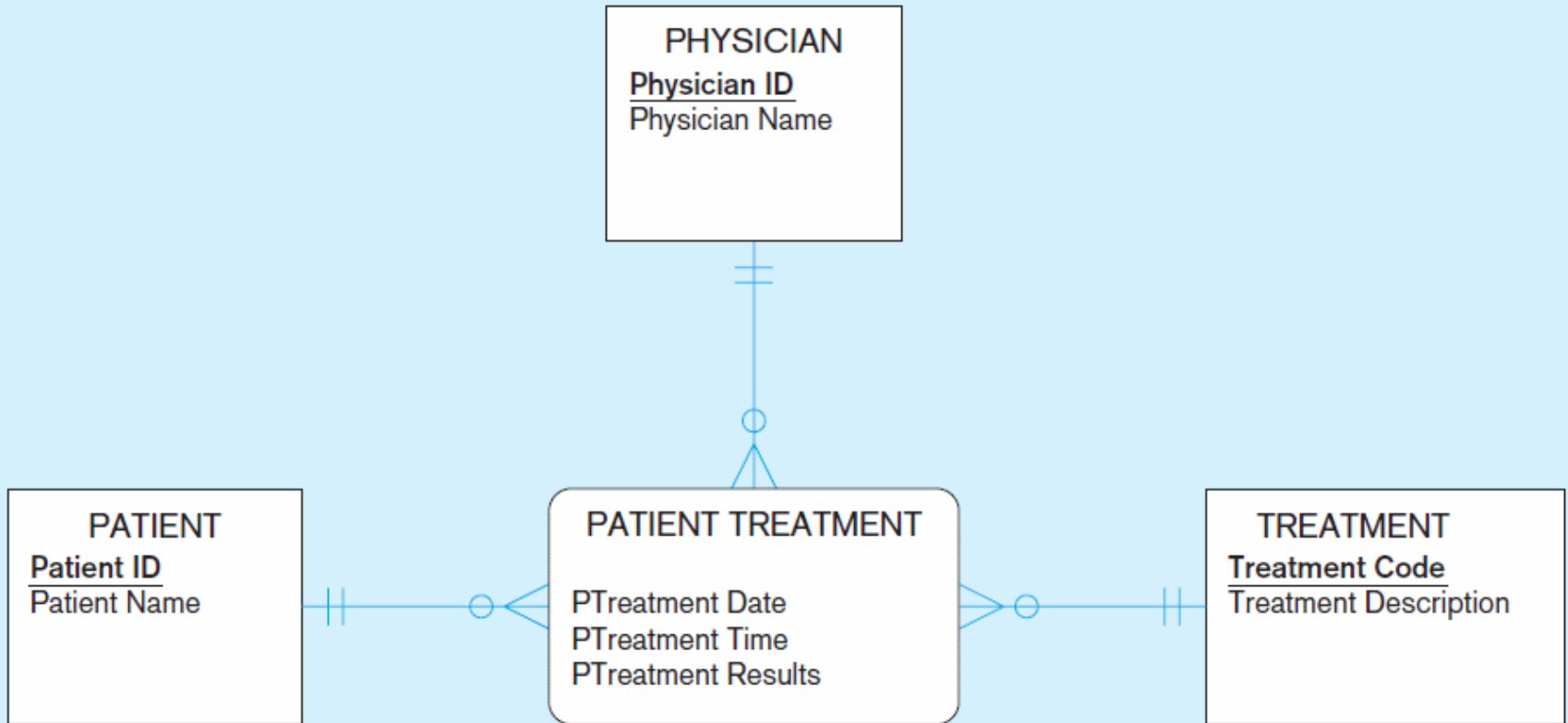
## Mapping Ternary (and n-ary) Relationships

- + One relation for each entity and one for the associative entity
- + Associative entity has foreign keys to each entity in the relationship



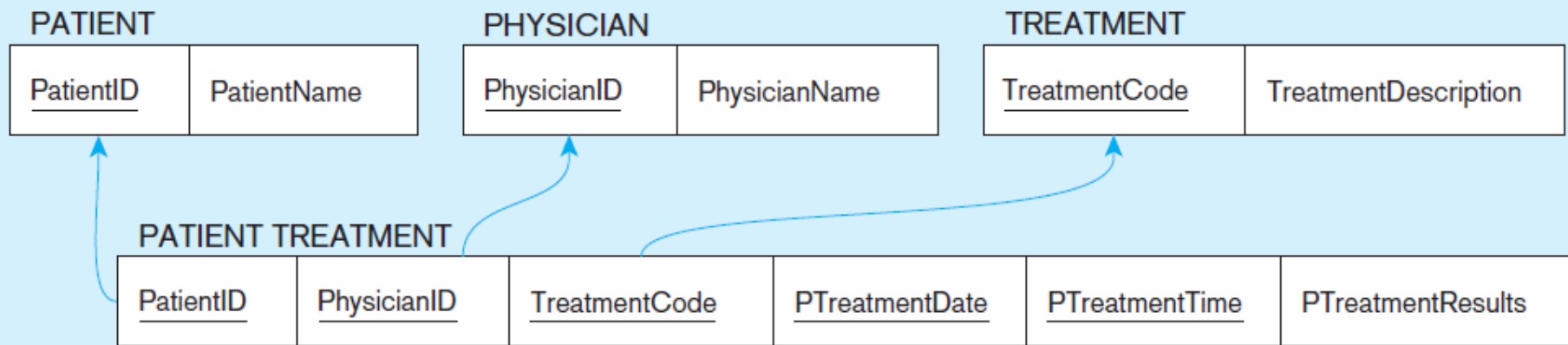
Figure 4-19 Mapping a ternary relationship

a) PATIENT TREATMENT Ternary relationship with associative entity



## Figure 4-19 Mapping a ternary relationship (cont.)

### b) Mapping the ternary relationship PATIENT TREATMENT



Remember that the primary key **MUST** be unique.

This is why treatment date and time are included in the composite primary key.

But this makes a very cumbersome key...

It would be better to create a surrogate key like Treatment#.

# TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Supertype/Subtype Relationships

- + One relation for supertype and for each subtype
- + Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- + Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- + 1:1 relationship established between supertype and each subtype, with supertype as primary table



Figure 4-20 Supertype/subtype relationships

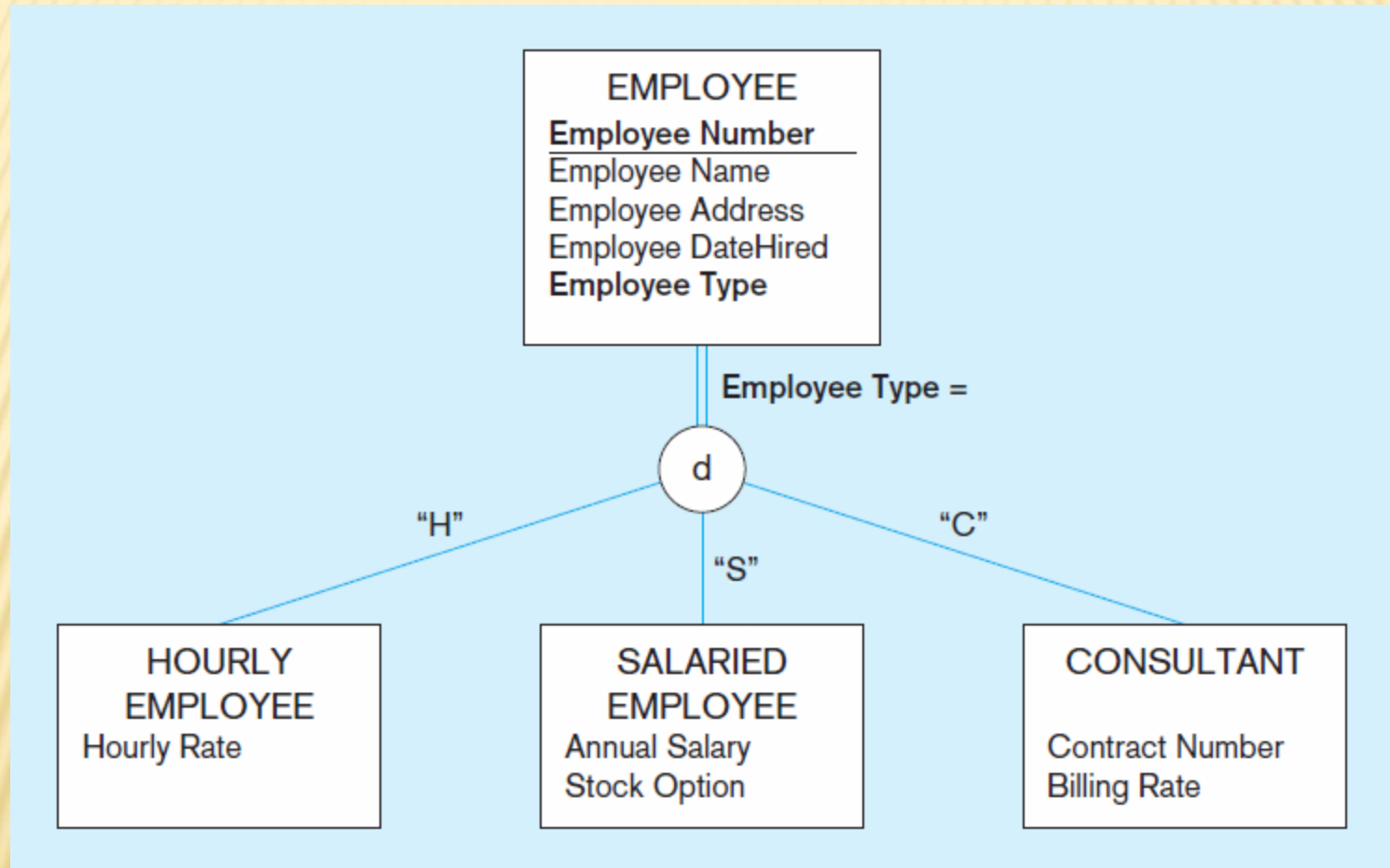
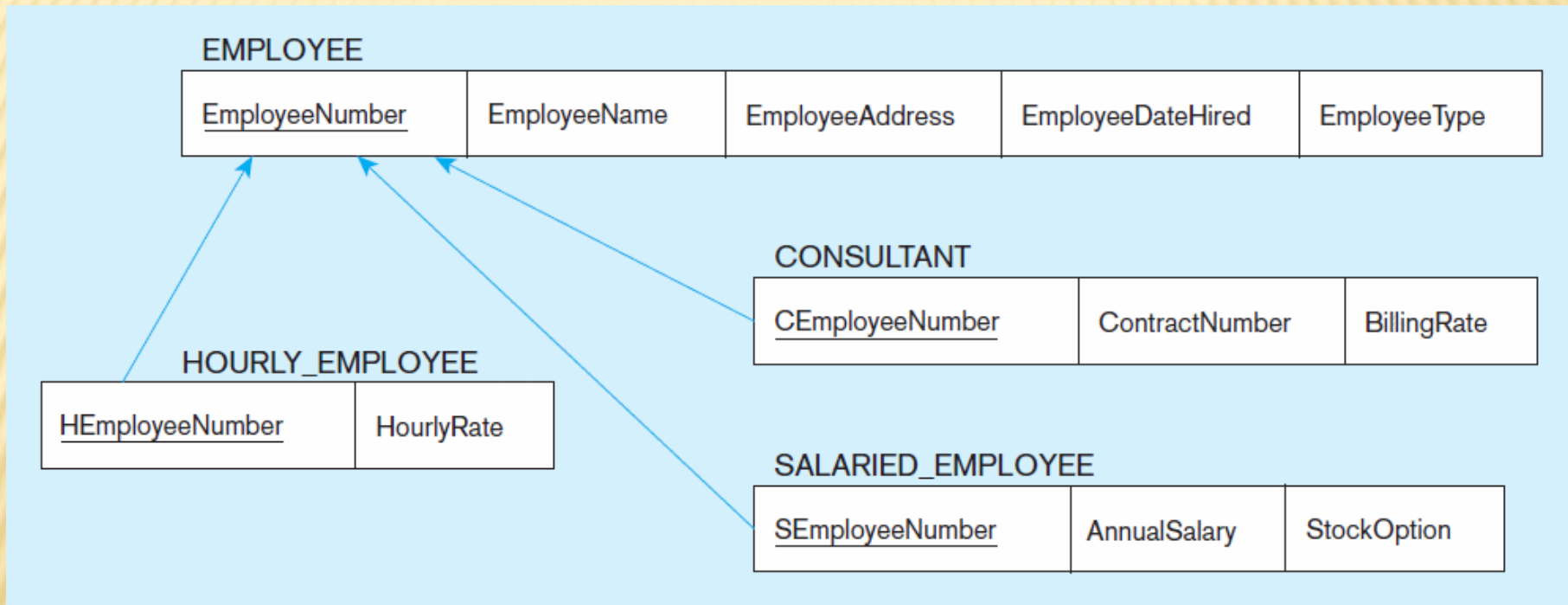


Figure 4-21

## Mapping supertype/subtype relationships to relations



These are implemented as one-to-one relationships.

# DATA NORMALIZATION

---

- ✘ Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that ***avoid unnecessary duplication of data***
- ✘ The process of decomposing relations with anomalies to produce smaller, ***well-structured*** relations



# WELL-STRUCTURED RELATIONS

---

- ✗ A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- ✗ Goal is to avoid anomalies
  - + **Insertion Anomaly**—adding new rows forces user to create duplicate data
  - + **Deletion Anomaly**—deleting rows may cause a loss of data that would be needed for other future rows
  - + **Modification Anomaly**—changing data in a row forces changes to other rows because of duplication

**General rule of thumb: A table should not pertain to more than one entity type.**

# EXAMPLE-FIGURE 4-2B

EMPLOYEE2

<u>EmpID</u>	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/201X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/201X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/201X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/201X
110	Chris Lucero	Info Systems	43,000	C++	4/22/201X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/201X
150	Susan Martin	Marketing	42,000	Java	8/12/201X

Question—Is this a relation?

Answer—Yes: Unique rows and no multivalued attributes

Question—What's the primary key?

Answer—Composite: EmpID, CourseTitle



# ANOMALIES IN THIS TABLE

---

- ✗ **Insertion**—can't enter a new employee without having the employee take a class (or at least empty fields of class information)
- ✗ **Deletion**—if we remove employee 140, we lose information about the existence of a Tax Acc class
- ✗ **Modification**—giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?

Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities.





**This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.**

Copyright © 2011 Pearson Education, Inc. Publishing as Prentice Hall